Kaggle Titanic

10636010 張佳裕
10636014 張榕真
10636024 張瑜倢

一、摘要

提出問題：研究的問題是什麼樣的人在泰坦尼克號中更容易存活？

下載數據：找到 Kaggle 網站的泰坦尼克專案中的 data 的頁面下載資料檔案。下載的文件有 3 個，分別如下所述：

1. gender_submission.csv: kaggle 給我們一個提交結果的案例檔，告訴我們預測完成後提交的檔要與這個檔的格式一樣
2. test.csv 文件：測試資料使用機器學習模型對生存率進行預測,並將預測結果提交給 kaggle，讓 Kaggle 來評估我們的模型的正確率
3. train.csv 文件：裡面的資料為訓練資料，用於建立模型

二、介紹(研究背景與研究目的)

1. 研究背景

Titanic 的沉沒是歷史上最悲慘的沉船事件之一。1912 年 4 月 15 日，在她的處女航中，Titanic 與冰山相撞後沉沒，2224 名乘客和機組人員中造成 1502 人死亡。

這場聳人聽聞的悲劇震驚了國際社會，因而也促成了更好的船舶安全規定。

造成海難失事的原因之一，是乘客和機組人員沒有足夠的救生艇，但其實有些人因為身分的緣故生存率高於他人，比如女人，孩子和上流社會人士。

2. 研究目的

預測 Titanic 乘客生存與否?

- Input: 乘客屬性，包括姓名、性別、年齡、登船港口、客艙等級、客艙號、船上兄弟姊妹數量+配偶數量、父母小孩的數量、船票價格、乘客編號、船票編號
- Output: 乘客生存與否

依據 Input 所得知的資訊，來預測各個乘客是否會在 Titanic 沉船的意外中生存下來。

三、資料集介紹、資料集來源及資料預處理

1. 資料集來源

Kaggle 競賽: Titanic: Machine Learning from Disaster

https://www.kaggle.com/c/titanic/data

2. 資料集介紹

	欄位定義	備註
PassengerId	資料編號	
Survived	是否生存	0 = 死亡, 1 = 生存
Pclass	艙等	1 = 最高,3 = 最低
Name	姓名	
Sex	性別	
Age	年齡	
SibSp	是否有兄弟姊妹/配偶於船上	
Parch	是否有父母/子女於船上	
Ticket	票號	
Fare	票價	
Cabin	船艙號碼	
Embarked	出發港口	C = Cherbourg Q = Queenstown S = Southampton

3. 資料特徵及預處理

從資料集當中可以發現性別與生存與否有著顯著的關聯，因此以此性別欄為始做分析特徵處理。

預處理

- 性別資料原為文字型態的 male、female 轉換為 0, 1，以方便程式分析時使用

```
# Convert Sex
df_data['Sex_Code'] = df_data['Sex'].map({'female' : 1, 'male' : 0}).astype('int')
```

票價和艙等都是屬於彰顯乘客社會地位的一個特徵，而從資料中發現買較貴票價(艙等)的人，生存率較高，故將此兩欄位加入作為分析條件。然而，由於票價在資料中是以連續數字表示，因此我們需要將票價切分成幾個區間。

預處理

- 由於在票價欄位中出現缺失值，但僅有一項，因此先填入中位數

```
# Filling missing values
df_data['Fare'] = df_data['Fare'].fillna(df_data['Fare'].median())
```

- 嘗試將資料分等，並得出資料切分成 5 份分數最高

```
# Making Bins
df_data['FareBin_4'] = pd.qcut(df_data['Fare'], 4)
df_data['FareBin_5'] = pd.qcut(df_data['Fare'], 5)
df_data['FareBin_6'] = pd.qcut(df_data['Fare'], 6)

label = LabelEncoder()
df_data['FareBin_Code_4'] = label.fit_transform(df_data['FareBin_4'])
df_data['FareBin_Code_5'] = label.fit_transform(df_data['FareBin_5'])
df_data['FareBin_Code_6'] = label.fit_transform(df_data['FareBin_6'])

# cross tab
df_4 = pd.crosstab(df_data['FareBin_Code_4'],df_data['Pclass'])
df_5 = pd.crosstab(df_data['FareBin_Code_5'],df_data['Pclass'])
df_6 = pd.crosstab(df_data['FareBin_Code_6'],df_data['Pclass'])

display_side_by_side(df_4,df_5,df_6)

# plots
fig, [ax1, ax2, ax3] = plt.subplots(1, 3,sharey=True)
fig.set_figwidth(18)
for axi in [ax1, ax2, ax3]:
    axi.axhline(0.5,linestyle='dashed', c='black',alpha = .3)
g1 = sns.factorplot(x='FareBin_Code_4', y="Survived", data=df_data,kind='bar',ax=ax1)
g2 = sns.factorplot(x='FareBin_Code_5', y="Survived", data=df_data,kind='bar',ax=ax2)
g3 = sns.factorplot(x='FareBin_Code_6', y="Survived", data=df_data,kind='bar',ax=ax3)
# close FacetGrid object
plt.close(g1.fig)
plt.close(g2.fig)
plt.close(g3.fig)
```

資料集中發現了乘客持有相同的船票意味著他們可能是家人或是朋友，而在訓練集上這些互相有連結的人常常是一起活下來或是一起喪命，因此先將關係欄位(父母、子女、兄弟姊妹、配偶)設維新欄位以進行分析有多少持有同票根的人是親屬或只是朋友。

預處理

- 將持有相同票根的資料匯出為新的資料表，並顯示姓名、票價、艙位、家庭人數
- 透過家庭成員人數這個特徵來分類，是否有親屬關係
- 從資料中過濾出重複的票根，並且如果該票根群組中有人生還 則定義新欄位 Connected_Survival = 1 ；沒有人生還，則定義 Connected_Survival = 0 ；剩下的沒有生還資訊，定義 Connected_Survival = 0.5

```
# the same ticket family or friends
df_data['Connected_Survival'] = 0.5 # default
for _, df_grp in df_data.groupby('Ticket'):
    if (len(df_grp) > 1):
        for ind, row in df_grp.iterrows():
            smax = df_grp.drop(ind)['Survived'].max()
            smin = df_grp.drop(ind)['Survived'].min()
            passID = row['PassengerId']
            if (smax == 1.0):
                df_data.loc[df_data['PassengerId'] == passID, 'Connected_Survival'] = 1
            elif (smin==0.0):
                df_data.loc[df_data['PassengerId'] == passID, 'Connected_Survival'] = 0
#print
print('people keep the same ticket: %.0f '%len(deplicate_ticket))
print("people have connected information : %.0f"
      %(df_data[df_data['Connected_Survival']!=0.5].shape[0]))
df_data.groupby('Connected_Survival')[['Survived']].mean().round(3)
```

started 21:22:11 2018-06-16, finished in 2.94s

最後則發現生存率與年齡也有明顯的關聯，小孩的生存率可能比成年人還要高，因此對年齡欄位進行分析。但這個特徵中我們會面臨 20%缺失值的問題，故必須先將缺失值填滿。

預處理

- 先將姓名中的稱謂分為五類，Mr.、Master、Miss、Mrs.、其他稱謂
- 使用姓名中的稱謂來填補年齡的缺失值，而各稱謂所代表的年齡則利用分類中所有資料的中位數計算出。(Mr.- 29 歲，Master 4 歲，Miss- 22 歲，Mrs. - 36 歲，其他稱謂- 47 歲。)

```
# extracted title using name
df_data['Title'] = df_data.Name.str.extract(' ([A-Za-z]+)\.', expand=False)
df_data['Title'] = df_data['Title'].replace(['Capt', 'Col', 'Countess', 'Don',
                                             'Dr', 'Dona', 'Jonkheer',
                                             'Major','Rev','Sir'],'Rare')
df_data['Title'] = df_data['Title'].replace(['Mlle', 'Ms','Mme'],'Miss')
df_data['Title'] = df_data['Title'].replace(['Lady'],'Mrs')
df_data['Title'] = df_data['Title'].map({"Mr":0, "Rare" : 1, "Master" : 2,"Miss" : 3, "Mrs" : 4 })
Ti = df_data.groupby('Title')['Age'].median()
Ti
```

started 20:08:34 2018-06-16, finished in 29ms

四、機器學習或深度學習方法(使用何種方法)

Base Mode:在我們開始測試各式各樣的特徵之前，我們必須先選一個模型，當然我們盡量選用對抗噪聲較強的模型(SVM，KNN，隨機森林)，基本上我的考量是這樣的:

1. 廣泛性

鐵達尼號的資料集非常小僅有 891 筆資料，因此 SVM 及 KNN 較為人詬病的問題"在較大的資料集非常沒有效率"也會因為資料集小獲得紓解，然而，隨機森林由於其平行化計算的特質在資料集小或是大時的運算效能都不錯。因此在廣泛性上隨機森林得一分。

2. 預處理

資料在餵進模型之前必須先經過預處理過程，SVM 及 KNN 是以距離為基本來做超平面切分/鄰近投票，其中的手續比較多，也增加處理不好的風險，另一方面，隨機森林是以不純度函數來切分樣本，因此不需要歸一化或標準化，簡化了建模時的步驟。

如果是想要使用 KNN 及 SVM 的朋友，目前最大眾使用的是 StandardScaler(扣掉平均值後除以標準差)以及 MinMaxScaler(除以最大最小值)，在 Kaggle 上的討論區上有人測試過兩種，基本上是 case by case，可能兩者都試試看，來看哪個可以有比較好的 LB Score，不過如果要比較優劣的話:

(1) StandardScaler

- 好處:處理完之後皆符合常態分佈(平均值=0，標準差=1)，這對大部分的演算法都是好的。
- 壞處:如果你的特徵原本是很多 0 少數 1 的那種(sparse)，經過處理之後就不 sparse 了，一旦資料量一大，運算較快的優勢就會消失。

(2) MinMaxScaler

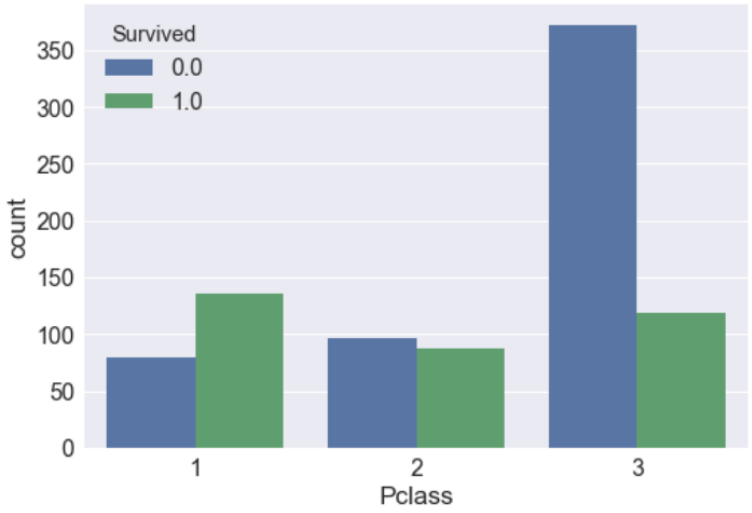
- 好處:如同上所說，可以保持特徵的 sparsity 同時控制每個特徵維持相等的權重。
- 壞處:由於處理完之後並沒有變換分佈，必須謹慎地確認演算法是否會因為資料分佈非高斯就表現得很差。

五、研究結果及討論(含模型評估與改善)

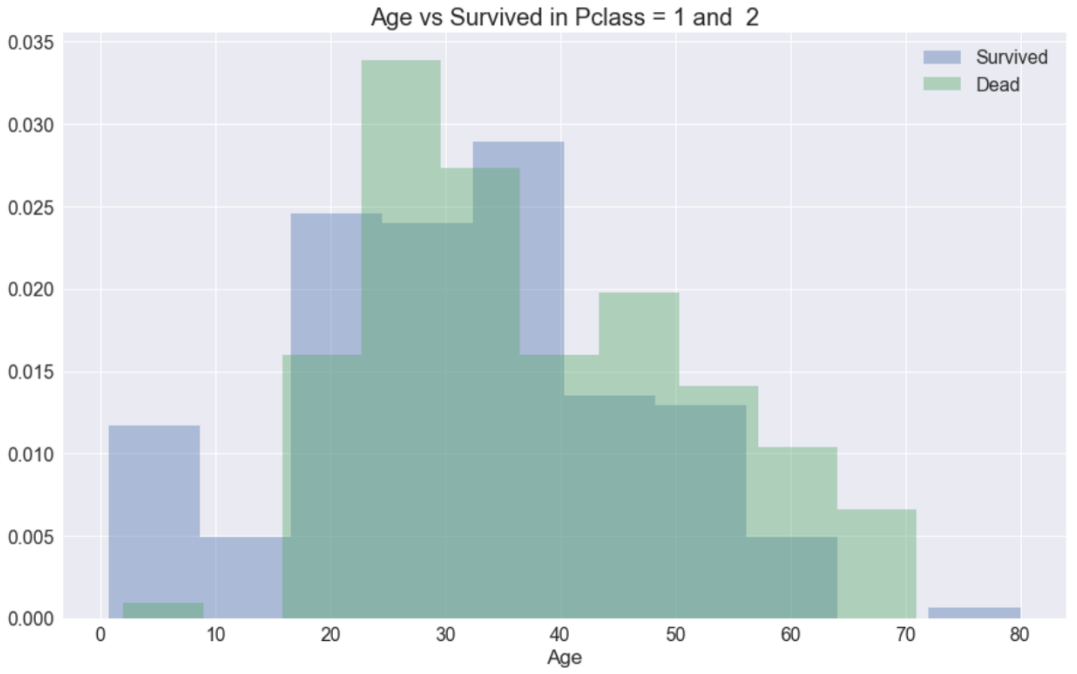
使用訓練集的資料作為輸入，進行模型分析後，我們得知，乘客存活率與性別、年齡、客艙等級以及同行者有關。

相較於男性(存活率 18%)，女性存活率(存活率 75%)更高，因為普遍男性較紳士，因此讓多數女性先搭乘救生艇逃生。

	Pclass	Survived
0	1	0.630
1	2	0.473
2	3	0.242



而年齡小於 16 歲的青少年存活率更高，據觀察，大人普遍愛子心切，因此傾向先讓孩童搭乘救生艇。

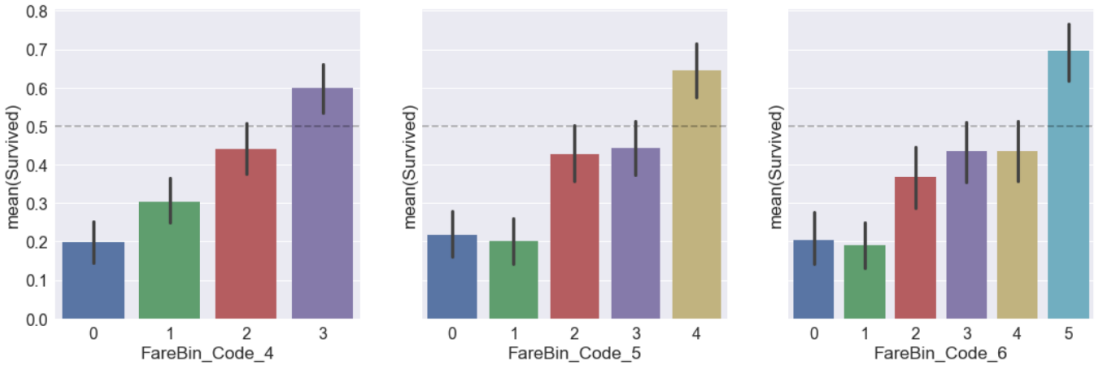


若乘客購買較高票價艙等的船票，會相較於其他低票價搭乘者較快速接收到即將沈船的通知，因而也能延長準備逃生的時間。

Pclass	1	2	3
FareBin_Code_4			
0	8	6	323
1	0	128	193
2	77	104	147
3	238	39	46

Pclass	1	2	3
FareBin_Code_5			
0	8	6	261
1	0	36	218
2	0	124	132
3	95	99	71
4	220	12	27

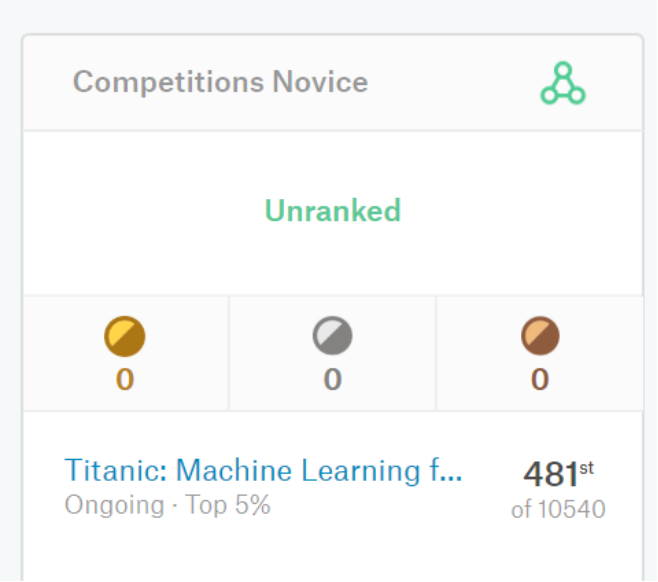
Pclass	1	2	3
FareBin_Code_6			
0	8	6	222
1	0	0	218
2	0	128	76
3	14	83	128
4	118	48	46
5	183	12	19



透過比對姓名、家人同行數量、配偶同行數量以及朋友同行數量(合稱關係人數)得知，若獨自乘船的乘客，在逃生時，可能無法取得他人協助，因而存活率相對於同行者較多的乘客低。

	Survived
Connected_Survival	
0.0	0.225
0.5	0.298
1.0	0.728

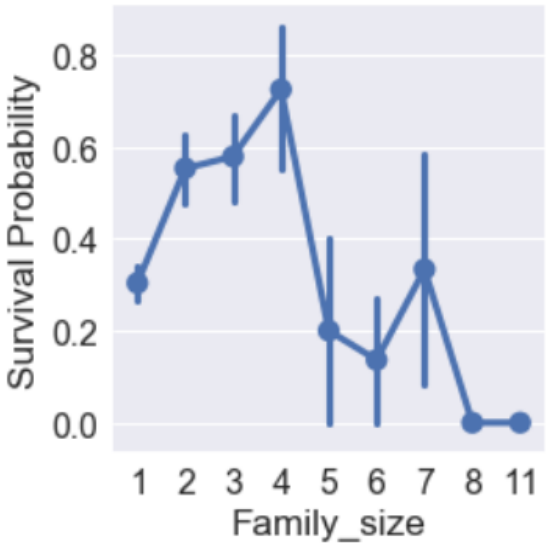
六、結論



在鐵達尼號的資料集裡，先以性別與艙等作為基礎特徵獲得 0.76555 的準確率並以此為基準，加入票價(Fare)、連結(Connected_Survival)、年齡(Age)、家庭數(Family_size)等特徵得以下結論。

```
g = sns.factorplot(x='Family_size', y='Survived',data=df_data)
g = g.set_ylabels("Survival Probability")
```

started 13:44:27 2018-06-17, finished in 632ms



我們可以看到獨自一人的生存機率低，2~4 人的家庭生還機率比較高，之後隨著人數增加而遞減，這時候我們就可能會腦補:獨自一人可能逃生時沒有受到幫助，所以生還率低，2~4 人可以互相幫忙，但是人太多時可能又會因為無法找齊全部的家人逃生時有猶豫等等，而將其分成 3 類，獨自一人(0)，2~4 人(1)，5 人以上(2)，分析如下:

```
# cut into 3 class
df_data['L_Family'] = df_data['Family_size'].apply(lambda x: 0 if x<= 4 else 1).astype(int)
df_data.loc[ df_data['Family_size'] == 1, 'FamilyClass'] = 0
df_data.loc[ (df_data['Family_size'] <= 4) & (df_data['Family_size'] > 1), 'FamilyClass'] = 1
df_data.loc[ df_data['Family_size'] >= 5, 'FamilyClass'] = 2
df_data['FamilyClass'] = df_data['FamilyClass'].astype(int)
df_data[['FamilyClass','Survived']].groupby(['FamilyClass']).mean()
```

started 13:52:45 2018-06-17, finished in 44ms

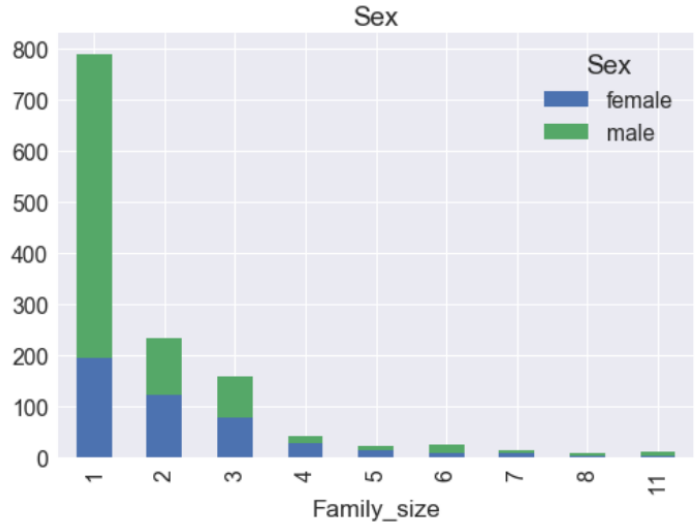
FamilyClass	Survived
0	0.303538
1	0.578767
2	0.161290

然而，這有可能只是我們的空想，我們來看看家庭人數對性別的分析圖形。

```
# display(pd.crosstab(df_data['Sex'],df_data['Family_size'],margins=True))
# visualize
pd.crosstab(df_data['Family_size'],df_data['Sex']).plot(kind='bar',stacked=True,title="Sex")
```

started 14:26:52 2018-06-17, finished in 398ms

```
<matplotlib.axes._subplots.AxesSubplot at 0x25a3d1685c0>
```



我們看家庭人數=1，真的是因為獨自一人沒有受到幫助而導致較低的生存率嗎?**有可能，但是不見得**，上表中我們可以清楚看到家庭人數=1 之中有接近將近 75%的男性，男性的生存率本來就低，FamilyClass=0 很有可能僅僅是性別特徵的一個重複而已!

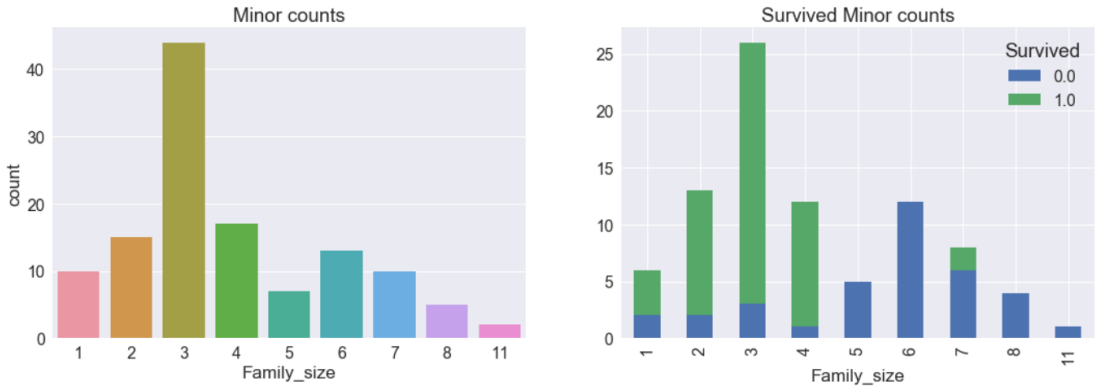
然而家庭人數 2~4 呢?是否真的是因為互相幫忙導致的生存率提高，還是又是其他特徵的投射而已? 我們來看家庭人數對小孩數量的分析。

下圖的左圖是家庭人數中小孩個數的統計，右圖是家庭人數中小孩存活的個數統計。

```
Minor_mask = (df_data.Ti_Minor == 1)
fig, [ax1, ax2] = plt.subplots(1, 2)
fig.set_figwidth(18)
sns.countplot(df_data[Minor_mask]['Family_size'],ax=ax1)
ax1.set_title("Minor counts")
pd.crosstab(df_data[Minor_mask]['Family_size'],df_data[Minor_mask]['Survived'])\
.plot(kind='bar',stacked=True,title="Survived Minor counts",ax=ax2)
```

started 14:56:56 2018-06-17, finished in 606ms

```
<matplotlib.axes._subplots.AxesSubplot at 0x25a40f2fd68>
```



在家庭人數=2,3,4 中，除了小孩的個數較高之外，存活的比例也非常高，這意味著家庭人數=2~4 人的生存率較高很有可能僅僅是小孩存活的多，這已經涵蓋在 Minor 的特徵當中!

至於剩下的 Family_size＞5 人呢?從分析中並沒有看出和其他特徵相關的影子，但如果我們僅是猜測其生還率低的原因是因為找不到其他家人猶豫而降低了生還率，這個理由我們在連結(Connected_Survival)已經做過同樣的推測，若僅是因為這個理由多造一個特徵，那確確實實又是冗餘了!

七、參考文獻

1. https://chtseng.wordpress.com/2017/12/24/kaggle-titanic%E5%80%96%E5%AD%98%E9%A0%90%E6%B8%AC-1/
2. https://zhuanlan.zhihu.com/p/31743196
3. https://ithelp.ithome.com.tw/articles/10206089
4. https://medium.com/@yehjames/%E8%B3%87%E6%96%99%E5%88%86%E6%9E%90-%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E7%AC%AC4-1%E8%AC%9B-kaggle%E7%AB%B6%E8%B3%BD-%E9%90%B5%E9%81%94%E5%B0%BC%E8%99%9F%E7%94%9F%E5%AD%98%E9%A0%90%E6%B8%AC-%E5%89%8D16-%E6%8E%92%E5%90%8D-a8842fea7077
5. http://yourgene.pixnet.net/blog/post/118394064-%E6%8E%A2%E7%B4%A2%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92%E5%8F%8A%E8%B3%87%E6%96%99%E7%A7%91%E5%AD%B8%E7%9A%84%E5%A5%BD%E5%9C%B0%E6%96%B9-kaggle
6. https://github.com/minsuk-heo/kaggle-titanic
7. https://github.com/ahmedbesbes/How-to-score-0.8134-in-Titanic-Kaggle-Challenge
8. https://github.com/topics/titanic-dataset
9. https://github.com/antonfefilov/titanic
10. https://github.com/prachi1210/titanic-survival
11. https://github.com/rebeccabilbro/titanic
12. https://medium.com/@yulongtsai/https-medium-com-yulongtsai-titanic-top3-8e64741cc11f
13. https://www.kaggle.com/francksylla/titanic-machine-learning-from-disaster
14. https://www.kaggle.com/tjsauer/titanic-survival-python-solution
15. https://zhuanlan.zhihu.com/p/28586467
16. https://www.kaggle.com/sgus1318/titanic-analysis-learning-to-swim-with-python